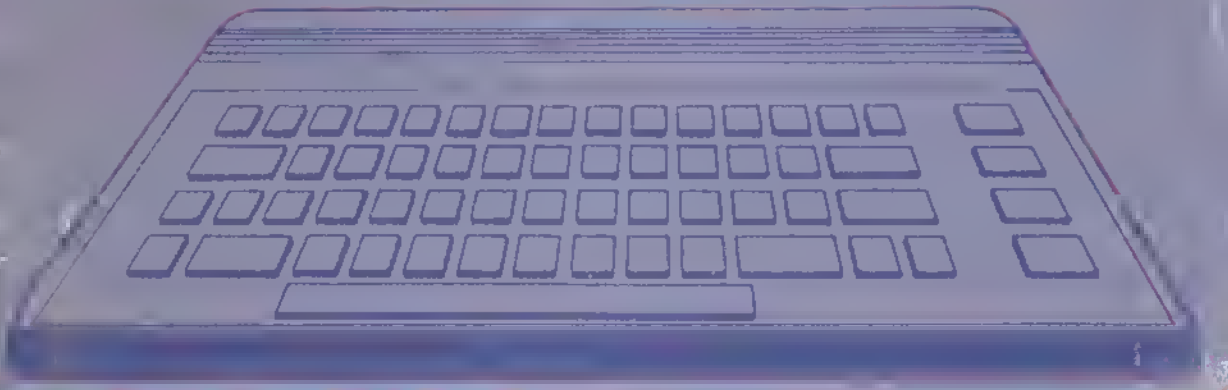
STACK

ARROW

a cartridge for the

COMMODORE 64



popular computer accessories

INTRODUCTION

Arrow and Arrow Plus are both utility cartridges for the CBM 64. The Arrow firmware is contained in a 4K Eprom, while the Arrow Plus firmware is contained in two 4K Eproms.

Arrow features cassette operations (Load, Save, Verify and Append) all at up to 7 times normal speed. Arrow also features a machine language monitor, hexadecimal calculator and the commands to use the powerful Arrow Plus assembler firmware.

Arrow Plus features all the functions of Arrow, but in addition to this it contains a full symbolic two pass assembler, and disassembler supplied on a supplementary cassette.

This manual is split into two sections, the first section contains information on the features that are common to both Arrow and Arrow Plus. While the second section contains instructions for the operation of the Arrow Plus assembler, and disassembler

INSTALATION of ARROW / ARROW PLUS

1. Ensure that the CBM 64 is turned off.
2. Insert your cartridge into the cartridge expansion slot with the label upper most.
3. Turn on the CBM 64.

   Both of the cartridges are self-starting so the extra commands are available each time you turn on the CBM 64.

N.B.

Serious damage may occur to both the Arrow/Arrow Plus cartridges and the CBM 64 if any attempt is made to insert a cartridge with the CBM 64 powered up.,

SECTION ONE: FEATURES COMMON TO BOTH ARROW and ARROW PLUS.

HIGH SPEED CASSETTE OPERATION

The four commands ←-S ←-L ←-V and ←-A are used to SAVE, LOAD, VERIFY or APPEND at high speed (3600 baud). All except the shortest of programs will be handled 6 to 7 times faster than the standard speed.
←-is used to indicate the key at the top left corner of the CBM 64 keyboard.
Program names are limited to 16 characters.
Except for a program save operation, the program name is optional and if the name specified is shorter than the name used when the program was saved, a match on the beginning of the file name (corresponding to your keyed input) is sufficient.

After each cassette Input/Output operation, the length of the program handled will be displayed on the screen.
←-S"PROGRAM NAME" is used ot save a BASIC program on the Datasette with a 4 second leader preceding the program. When using this command, the tape should be positioned on the magnetic portion of the tape and not on a long non-magnetic leader.
←-L"PROGRAM NAME" :- LOAD HIGH SPEED BASIC PROGRAM
←-T"PROGRAM NAME" is used to save a BASIC program on the Datasette with a 10 second leader preceding the program.

← A"PROGRAM NAME" is used to add a BASIC program from the Datasette to the BASIC program in the memory of the CBM 64. The append function works only for BASIC programs and the program appended will be added to the end of the existing program in memory without regard to line numbers.

← V"PROGRAM NAME" is used to verify that the BASIC program on your Datasette matches the program in memory. In the event that it does not, ?? will be displayed. Note that this is also a method of skipping a program on the Datasette for example to save another program after a particular program on tape. Note that ←-V will always display ?? when used after ← A.

← S"PROGRAM NAME",SSSS,EEEE (SSSS=start address and EEEE=end address plus 1) is used to save a machine language program or block of memory with a 4 second leader. The ← T form of this instruction may be used to save with a 10 second leader.

To load or verify a machine language program, use SHIFT of the L and V keys. ← [SHIFT]L or ←-[SHIFT]V

TAPE POSITIONING

The command ← Pd where d is digit from 1 to 9 is used to advance the cassette to one of nine positions using the Fast Forward button. Each block skipped is sufficient for a 16K program.

HEXADECIMAL CALCULATOR/CONVERTOR.

You may enter this mode by typing ←-H which will cause two counters to appear on the bottom line of the screen. At first they will both be zero, but if you type in hexadecimal number you will see it appear in the right-hand counter while its decimal equivalent is displayed in the left-hand counter.

You can add or subtract using the plus, minus and equals keys and you can switch from hexadecimal to decimal input or back again by using the * key. The H or D displayed at the end of the bottom line indicates which form of input, hexadecimal or decimal is currently in use.

Repetition of the equals key causes the last addition or subtraction to be repeated.

For Multiple conversions from decimal to hexadecimal or vice-versa enter the equals sign after keying in the number to be converted. Remember that the entry mode is indicted at the end of the bottom line.

The largest decimal value that can be handled is 65,535 with the hexadecimal maximum at $FFFF.

To exit from this mode type X.

DEACTIVATION

If, for some particular reason, you need to disable the ARROW functions,←O(oh, not zero) may be used. You can reactivate by keying in a SYS32810.

TEXT EOITING

There is a FIND command with an optional replacement field were both the search string and the replacement string may have up to 20 characters each. The format is :-

←F" search string"/replacement string/

If you want to search and replace with nothing (delete the search string), use the left arrow key (←) as the replacement string.

If there is no replacement string, a match will result in the old line being displayed with the new line repeated just below the old line with the replacement made, including memory. If you do not desire this replacement, you simply use the cursor Up key to get the old line and press return. If the replacement was correct, nothing has to be done.

AUTOMATIC LINE NUMBERING

←N first line number, increment may be used to initiate automatic line numbering. To exit from AUTO type←Q.

LINE RENUMBERING

←R first line number, increment may be used to renumber the lines in an assembler source program being prepared or modified for the assembler. This is not intended for BASIC programs and therefore does not take into account GOTO,GOSUB, THEN and ON N GOTO statements. The default values are first line number 10 with an increment of 10.

LINE DELETION

←D first line number, last line number may be used to delete block of lines. N.B. Do not use without specifing the line range assembler. (see Arrow + section of manual for further information)

←E initiates the execution of the assembler. (Arrow + required).

←B initiates execution of assembler at the starting address plus 5 when user coded modification of the output character stream is used. For use with certain printers. Designed for use with the Vic Printer, (Arrow + required)

MOVE MEMORY

←M address, end address, new start address may be used to move block of memory. The original block is not modified unless there is a conflict between the starting address and the new starting address. Note that code is merely moved and logically relocated. Immediately after each byte is moved, the byte is compared to memory and if there is a difference due to a move ROM or a bad RAM address, the contents and the address will be displayed on the screen.

COMPARE MEMORY

←C beginning address, end address, second beginning address may be used to compare any two blocks of memory. Any difference will be displayed on CRT.

MACHINE LANGUAGE MONITOR

←X may be used to go onto the machine code monitor. The commands available are:-
M BBBB EEEE (beginning and end address in hex) to display a block of memory, 4 bytes per screen line. If EEEE is not specified, 4 bytes will be displayed. Moving the cursor and changing displayed bytes then keying a return will result in a change of the contents of memory.

←G BBBB will result in a jump to the specified address. If no address is specified, the contents of the PC will be used.

R results in a register display. The zones are defined as follows PC is the program counter, SR is the status register. AC is the accumulator, XR and YR are the registers X and Y and SP is the stack pointer. Changing these fields (always follwed by a return) will result in the new values being used upon a G instruction. When a BRK instruction (hex 00) is inserted in a machine language program, the monitor will be entered with the current values of all the above registers displayed.

S" file name",01,BBBB,EEEE (beginning address and end address plus one) may be used to write a block of memory in the standard (slow non-Arrow) format.

L" file name",may be used to load a block of memory saved via the S command above.

X may be used to return to normal BASIC operation (READY message).

SECTION TWO: 651Ø (65xx series) ASSEMBLER -ARROW PLUS ONLY.

The purpose of an assembler is to convert source code (mnemonic code) into the native language (machine code) of a particular type of computer. Using an assembler alleviates the programmmer of tedious address calculations and looking up operation codes etc. Source code can be easily modified at any time and the program re-assembled without having to re- calculate everything as when coding is pure machine code.

The source code for the assembler is keyed in line by line, in the same manner as a BASIC program. this means that SAVE-ing, and LOADing and modification of your source program may be done as if the source were BASIC program. (See example in Appendix).

The object code generated by this assembler is temporarily held in the top portion of memory (last address equal to the

content of $37/3B minus one) and occupies a multiple of 1024 bytes as defined when you initiate execution (see section on Execution) and may be moved during an assembly. At the end of each assembly you will be asked whether not you wish to transfer the object code to its execution address.

You may request either a symbol table listing or a full cross reference were the labels used are listed in alphabetical order with there associated value and each line number where the label was used. This is extremly useful when modifying a source program, particularly if the program is rather long.

SYNTAX

All source instructions are keyed in after the line number. There are five zones, separated by at least one space, as follows:-

1. Line Number. The line number may be any number from 1 to 63999. Note that the numbers keyed in (or obtained via ←-N) are for your use in editing source - the assembler listing will contain its own sequential line numbers beginning at 1.

2. Label. the label, if needed for the line, is keyed in right after the line number. It may contain from 1 to 9 alphanumeric characters but must not contain the plus or minus sign nor start with a period. A label may not match a mnemonic instruction. For example, LDA will not be considered as a label.
NOTE THAT ZERO PAGE LABELS MUST BE DEFINED BEFORE BEING USED.

3. Mnemonic or pseudo op code. The mnemonic instruction is selected from the following list and follows the MOS Technology standard:-

ADC	AND	ASL	BCC	BCS	BEQ	BIT	BMI	BNE
BPL	BRK	BVC	BVS	CLC	CLD	CLI	CLV	CMP
CPX	CPY	DEC	DEX	DEY	EOR	INC	INX	INY
JMP	JSR	LDA	LDX	LDY	LSR	NOP	ORA	PHA

PHP PLA PLP ROL ROR RTI RTS SBC SEC
SED SEI STA STX STY TAX TAY TSX TXA
TXS TYA.

See also the section on pseudo - op codes.

4. Operand. The operand also follows the MOS Technology standard. For immediate instructions or elements of a .B or .W instruction (see chapter of Pseudo Operations), the following conventions are used:-

a. A decimal number may be specified without a prefix.
b. The $ prefix specifies a hexadecimal number.
c. The greater than and less than symbols specify the high order or low order 8 bits of an expression.
d. The % sign followed by a string of up to eight 0's and 1's indicates a binary value.
e. The '(apostrophe) prefix may be used to define a one character ASCII code. By using multiple occurrences of this prefix in a .B statement you can enter messages to be displayed, for example.
f. Labels previously defined may be used in expressions including combinations such as label±label±decimal value.

5. Comments. The comments zone begins with a semi-colon character(;) and is used by the programmer to clarify a program listing. the assembler does not read this zone but merely outputs it as keyed.

N.B.

When using ARROW PLUS problems may occur, when using Absolute addressing these problems can be solved if the absolute address is defined as a label prior to using it.

```
i.e.    ADDR = $AABB
        LDA ADDR
rather than

        LDA $AABB.
```

PSEUDO OPERATIONS

All pseudo-op codes begin with a period and may contain up to four characters but only the first character is needed and only the first two are output.

.B may be followed by any number of operands or expressions separated by commas and results in one byte of object code per operand being assembled. If a label or label expression is used, the value assembled will be the low order eight bits after evaluation.

.W is similar to .B except that each expression results in a double byte result with the low order byte stored first.

.D followed by an expression is used to define a block of memory if any length. Note that the memory defined is not initialized to any particular value.

.O followed by an expression (any symbols used must have been previously defined) may be used to set the location counter (memory address) to whatever value you desire. You may have several .O statements in program but the lowest value must occur first and the overall length of the program, including any gaps, must fit within the object code buffer defined at execution time.. Also, the last .O instruction must be the highest value used for a program counter change.

.L is used to list the following lines on the output device . When the program is loaded the list function is on. Assembly errors are always listed on the output date.

.X is used to cancel the list function on the output device for the following lines.

.S is used if a symbol table (symbol and their addresses) is to be output at the end of the program.

.C is used if a sorted cross reference is to be output at the end of the program. The cross-reference consists of the line number where each symbol isused preceded by the symbol name, in alphabetical order, and symbol value. If .C is specified as well as .S only the cross-reference will be output. Note that for very large source programs it may not be possible to enter the .C function since 4 bytes are added to the symbol table per reference.

PRINTED OUTPUT

When you are satisfied with the results of an assembly, you may specify output to the printer by simply keying in OPEN 4.4:CMD 4 before you begin execution. This changes the output device which is normally the screen to be directed to a printer as device number 4 on the IEEE-488 bus.(SERIAL)

This assembler outputs lines of up to 200 characters followed by a carriage return at the end of each line . In order for you to adapt non-standard printers, you can place a jump instruction in addresses $3F9-3FB to enter your own coding. Before each character is output, a call to the routine at the user vector $3F9-3FB is made with the character in the accumulator. To convert one character for example from a carriage return to a line feed, simply change the value in the accumulator and do a return (RTS) instruction. If you wish to add characters to the output you can accomplish this by doing a JSR $FFD2 with each character to be added in the accumulator and then a final return to the assembler.

Note that this vector can also be used to paginate the output, for example.

Note that if you are using the output vector (see chapter on Printer Output) you must use the ←B command rather than the standard ←E command to begin execution.

The first thing that will happen after you give the ←E or ←B command is that a message will appear asking how many K(Kilobytes)(multiples of 1024 bytes) you require for your object code to which you may answer from Ø to 9. This will reserve a block of memory with the end of the object code buffer equal to the top of memory ($37/38 minus 1).

During the first pass of the assembler nothing is displayed on the screen. The second pass results in the listing being output to the screen or printer. Note that the assembly process may be suspended indefinately by depressing any key one time and then resumed by pressing another key.

The assembler output is divided up into seven zones as follows:-

1. Address of object code.
2. Object code in hexadecimal.
3. Sequential line number beginning at 1.
4. Label.
5. Mnemonic or pseudo operation code.
6. Operand.
7. Comments. Note that comments are supressed if the operand is more than 27 characters long.

During an assembly, any of the following error messages may be displayed if there are any error in your source program;

1. Program counter expression error.
2. Duplicate label.
3. Format error.
4. Invalid mnemonic.
5. Invalid operand.
6. Object code overflow.
7. Branch out of range.
8. Too many symbols.
9. Invalid symbol.

You will normally have to correct your source code and re-assemble upon occurrence of any of these errors.
At the end of the assembly another message will occur at the bottom of the display screen asking you if you wish to move the object code to its execution address or not. Once you have the object code at its execution address, you may initiate execution via the G function of the built-in monitor or via SYS call from BASIC. You may save your object code on tape using the ARROW or built in monitor.

```
#B? 1

                    1           .C
1200                2           .OR $12ØØ
1834                3 FIELD     = $1834
ØØ8A                4 ZPAGE     = $8A
78ØØ                5 PART2     = $78ØØ
ØØ14                6 OFFSET    = 2Ø
                    7 ;***************
                    8 ;ADDRESSING MODES
                    9 ;***************
12ØØ  69F4          10          ADC $F4         ;IMMEDIATE
1202  6D3418        11          ADC FIELD       ;ABSOLUTE
1205  658A          12          ADC ZPAGE       ;ZERO PAGE
1207  0A            13          ASL A           ;ACCUMULATOR
120B  18            14          CLC             ;IMPLIED
1209  61BA          15          ADC (ZPAGE,X)   ;PRE-INDEXED IND
120B  718A          16          ADC (ZPAGE),Y   ;POST INDEXED  "
120D  758A          17          ADC ZPAGE,X     ;ZERO PAGE
120F  7D3418        1B          ADC FIELD,X     ;ABSOLUTE X INDEXED
1212  793418        19          ADC FIELD,Y     ;ABSOLUTE Y INDEXED
1215  9003          20          BBC *+5         ;RELATIVE
1217  6C3418        21          JMP (FIELD)     ;INDIRECT
121A  B68A          22          LDX ZPAGE,Y     ;ZERO PAGE IN-
                    23;                          DEXED BY Y
                    24****************
                    25;OPERAND SAMPLES
                    26;***************
121C  A9ØF          27          LDA 15          ;DECIMAL
121E  A915          2B          LDA $15         ;HEXADECIMAL
1229  A918          29          LDA  FIELD      ;HIGH ORDER 8 BITS
1222  A934          30          LDA  FIELD      ;LOW ORDER B BITS
1224  Ø9DA          31          ORA %11011010   ;BINARY
1226  9ØØ5          32          BCC LOOP+3      ;LABEL WITH DISPLACEMEN
122B  BØØE          33          BCS LOOP+OFFSET6.; COMPLEX LABEL
122A  4C3418        34          JMP FIELD
122D  Ø234FE        35          .BY 2, FIELD,$FE ;MULTIPLE BYTE
1239  544558        36          .BY 'T,'E,'X,'T ;ASCII VIA .BY
1234  341B          37          .WO FIELD       ;WORD
1236  457B          38          .WO $7845,1Ø    ; MULTIPLE BYTE
                                                  EXPRESSION
```

```
FIELD          1834  11   18   19   21   29
                     30   34   35   37
LOOP           122A  32   33
OFFSET         ØØ14  33
PART2          78ØØ
ZPAGE          ØØ8A  12   15   16   17   22

   Ø ERRORS, 148F FREE

MOVE Y/N? N

READY.
```

DEMONSTRATION OF ASSEMBLER.

```
100 ; ARROW PLUS
110 ; DEMONSTRATION DF ASSEMBLER OPERATING MODES
120 ;                       COMMENT
130 .O 16000                ;ORIGIN
140 ADR=456                 ;LA8EL
150 .C                      ;CROSSREF
160 TXA                     ;ORG IMPLIED
170 LDA 123                 ;O+1 IMMEDIATE DECIMAL
180 LDA %111                ;O+3 IMM BINARY
190 LDA $FF                 ;O+5 IMM HEX
200 LDA ADR                 ;O+7 A8SOLUTE
210 LDA ADR,X               ;O+10 A8S+X
220 LDA ADR,Y               ;D+13 A8S+Y
230 ZER=123
240 LDA ZER                 ;O+16 ZERO PAGE
250 LDA (ZER,X)             ;O+18 IX INDEXED INDIRECT
260 LDA (ZER),Y             ;O+20 IY INDIRECT INDEXED
270 LDA ZER,X               ;D+22 ZPG+X
280 LDY ZER,X               ;D+24 ZPG+X
290 LDX ZER,Y               ;O+26 ZPG+Y (ZPG+Y IS FOR LDX ONLY)
300 JMP ADR                 ;O+28 ABSOLUTE
310 JMP (ADR)               ;O+31 INDIRECT  (IND IS FOR JMP ONLY)
320 ROR ZER                 ;O+34 ZERO PAGE
330 ROR A                   ;O+36 ACCUMULATOR
```

```
340 ROR ADR                ;0+37 ABSOLUTE
350 .B 123                 ;0+40 DATA BYTE OR BYTES
360 .W 456                 ;0+41 DATA WORD OR WORDS
370 RTS                    ;0+43 REMEMBER THE RETURN
3B0 ;PRINTER MUST BE OFF AT POWER-UP
390 ;THEN OPEN 4,4:CMD4
400 ;←N 100,10             NUMBER
410 ;←Q                    QUIT NUMBERING
420 ;←R 100,10             RENUMBER
430 ;←B OR E


←B
K? B

          1 ;    ARROW PLUS
          2 ;    DEMONSTRATION OF ASSEMBLER OPERATING MODES
          3 ;         COMMENT
3E80      4      .O   16000     ;ORIGIN
01C8      5 ADR  =    456       ;LABEL
          6      .C             ;CROSSREF
3E80 8A   7      TXA            ;ORG IMPLIED
3E81 A97B 8      LDA 123        ;0+1 IMMEDIATE DECIMAL
3EB3 A907 9      IDA %111       ;0+3 IMM BINARY
3E85 A9FF 10     LDA $FF        ;0+5 IMM HEX
3E87 ADCB01 11   LDA ADR        ;0+7 ABSOLUTE
3E8A BDCB01 12   LDA ADR,X      ;0+10 ABS+X
3EBD B9CB01 13   LDA ADR,Y      ;0+13 ABS+Y
007B      14 ZER =    123
3E90 A57B 15     LDA ZER        ;0=16 ZERO PAGE
3E92 A17B 16     LDA (ZER,X)    ;0+1B IX INDEXED DIRECT
3E94 B17B 17     LDA (ZER),Y    ;0+20 IY INDIRECT INDEXED
3E96 B57B 1B     LDA ZER,X      ;0+22 ZPG+X
3E98 B47B 19     LDY ZER,X      ;0+24 ZPG+x
3E9A B67B 20     LDX ZER,Y      ;0+26 ZPG+Y )ZPG+Y IS FOR LDX ONLY
3E9C 4CCB01 21   JMP ADR        ;0+28 ABSOLUTE
3E9F 6CC801 22   JMP (ADR)      ;0+31 INDIRECT (IND IS FOR J MP
                                                        ONLY)
3EA2 667B 23     ROR ZER        ;0+34 ZERO PAGE
3EA4 6A   24     ROR A          ;0+36 ACCUMULATOR
```

```
3EA5 6EC801 25       ROR ADR            ;0+37 ABSOLUTE
3EA8 7B     26       .B 123             ;0+40 DATA BYTE OR BYTES
3EA9 C801   27       .W 456             ;0+41 DATA WORD OR WORDS
3EAB 60     28       RTS                ;0+43 REMEMBER THE END
            29 ; PRINTER MUST BE OFF AT POWER-UP
            30 ;THEN OPEN 4,4:CMD4
            31 ;←N 100,10    NUMBER
            32 ;←O           QUIT NUMBERING
            33 ;←R 100,10    RENUMBER
            34 ;←B OR E OR SYS45056 ASSEMBLE

ADR   01C8 11  12  13  21  22
           25
ZER   007B 15  16  17  18  19
           20  23
   0 ERRORS , 416B FREE

MOVE Y/N? Y
READY.
```

6502 DISASSEMBLER

Fully compatible with the ARROW PLUS cartridge for the Commodore 64, this program produces source code which may be modified and/or re-assembled.

The general strategy is to locate the object code to be disassembled as high as possible in memory remembering that the ARROW PLUS cartridge (which must be in place to execute the Disassembler) is located at $8000-9FFF which makes $8000 the highest possible top of memory address. The Disassembler is loaded (into $C000-CA62) from cassette via the command left arrow L (in shift).

The format to begin a disassembly is:
SYS49152:beginning address, end address, actual address (for .OR pseudo op code). Each of the address fields must be 4

hexadecimal digits and the end address is really the end address plus 1.
Note the use of a colon (:) after the SYS command - it is obligatory. The first pass of the disassembler creates a symbol table which is used during the second pass. During the first pass there is nothing shown on the screen and during the second pass you will see the source code being listed on the screen as it is generated. The source code is 100% compatible with EZASM and can therefore be saved, modified and assembled as you wish.
The labels created have a Z prefix for zero page fields, a J prefix for addresses jumped to, a B prefix for addresses branched to, a T prefix for a table within the module and an X prefix for addresses external to the module being disassembled. All labels have a numeric suffix which corresponds to the address, in hexadecimal.
After a first disassembly, you may want to introduce some of your own labels for clarity. With the exception of labels beginning with Z, J, B, T or X followed by hexadecimal numerics, you can create labels of up to 9 characters by entering, in the format of BASIC lines, your labels as follows:

line number LABEL=$HHHH where HHHH is actual address.
Zero page and external labels will be output at the beginning of the disassembly and other labels will appear within the disassembly.
A disassembly will normally need some touching up since address tables and text are often intermixed with object code and only the human brain is capable of sorting that type of thing out at this time. Nevertheless, this is a precious tool for progress which you have in object form only and you wish to understand or modify.

DISCLAIMER.

Whilst every effort has been made to provide a flexible, reliable and above all low-cost product STACK COMPUTER SERVICES LTD. wish to point out that no claim is made for complete compatibility with any other equipment or program. The information given is believed to be accurate but no liability can be accepte for the consequences of any error. Ours is a policy of continue development and we therefore reserve the right to alter the design of specifications without prior notice.

REQUEST FOR INFORMATION

In order to provide the user with as much support as is practical, we would appreciate it if any useful comments or hints could be forwarded in to writing to:-

PRODUCT DEVELOPMENT
STACK COMPUTER SERVICES LTD
290/298 DERBY ROAD,
BOOTLE,
LIVERPOOL,
L20 8LN.

ERRATA ARROW PLUS / ARROW.

Page 14 Demonstration of assembler continued form page 13. Line numbers 400 to 430. All commands are prefix by ←

```
ie.   400 ; ←N 100,10
      410 ; ←Q
      420 ; ←R 100,10
      430 ; ← B OR E
```

Page 15 Same as above arrows not clear due to printing error

```
ie.   31 ; ←N 100,10
      32 ; ←Q
      33 ; ←R 100,10
      34 ; ←B OR E
```

NOTE Line 34 should not be (or SYS45056).

Page 13 The listing demonstration of assembler is not a program and should not be executed. It is a number of written statements showing some common examples of assembler instructions and their format.